JUMPING APPLICATION SECURITY SYSTEM

Priority Claim

This application claims priority under 35 USC 119(e) and 120 from U.S. Provisional Patent Application Serial Nos. 60/419,312 and 60/419,288, both filed on October 16, 2002 and entitled "Jumping Application Security System" and "Mobile Application Morphing System And Method" respectively, both of which are incorporated herein by reference.

Field of the Invention

This invention relates generally to a jumping application security system and method and in particular to a jumping application provisioning system and method that may be implemented for jumping applications that execute on various devices.

Background of the Invention

In traditional computing systems, communication between computers is either code (a software application)or data (a file containing information)and there is no notion of a program moving between hosts while it is being executed. Thus, with a typical computing system, a person may execute a software application (e. g., Microsoft Word)on his own computer and then forward the results of the execution of the software application (e. g., a Word document)to another user. The other user may then view the Word document by executing his own copy of Microsoft Word. A user may also send another user an executable software application file that the other user may download and execute on his own computer. However, these traditional computing systems do not recognize a single instantiation of a software program that may be executed by one or more different computers in order to complete the execution of the software application.

A jumping application, sometimes also called a jumping app, a mobile application, a mobile app, or a mobile agent, is a computer software application/program, or part of a computer program that can physically move from one computer to another (between hosts)while it is being executed: A jumping application 's software may or may not have been

previously installed on a particular computers prior to the arrival of the jumping application. The jumping applications are said to *jump* from one computer to another computer and the process of jumping from one computer to another computer is also referred to as a *jump.*

The process of initiating a jump between computers is commonly known as a *dispatch.* Typically, each jumping application will carry with it an ordered list or tree of hosts which the jumping application must visit during its execution, and such a list or tree is called the jumping application's *itinerary.* The computers that can receive and dispatch jumping applications are called *hosts.* The collection of hosts, computer networks, and software which executes and supports the jumping applications, and the jumping applications themselves, is called the *jumping application system.*

A jumping application typically has at least two parts: the state and the code. The state of the jumping application contains all of the data stored, carried, used, and/or computed by the particular jumping application. The code of the jumping application is the set of computer instructions which the host computer is intended to carry out on behalf of the jumping application during the execution of the jumping application by the particular host computer. In addition, a jumping application may have other parts, including an Access Control List (ACL), an itinerary, a datastore, an audit log, etc. A jumping application 's software may or may not have been previously installed on the computers prior to the arrival of the jumping application.

Jumping applications have demonstrable benefits for computer systems. However, they also create security problems. In particular, a hostile host computer might tamper with the code, the state, or the configuration of a jumping application before dispatching it to another host, in order to attack that host or another part of the jumping application system. Thus, there is a need to ensure that a host computer cannot adversely alter the configuration of a jumping application.

Current implementations of jumping application systems support techniques to ensure that any code transmitted to a host computer comes from a location which is known (or believed)to be safe. This is accomplished by simply preventing any untrusted host from

transmitting any executable code. Current implementations are binary: either a host can transmit code to other hosts, or a host cannot transmit code to another host.

However, current jumping application implementations do not adequately handle the situation in which an untrusted host needs to specify the behavior of a jumping application on another host. Thus, it is desirable to provide a system which allows an untrusted host to specify the behavior of a jumping application when that jumping application is on another host and it is to this end that the present invention is directed.

Summary of the Invention

The security system and method in accordance with the invention allows an untrusted host to specify the behavior of a jumping application by describing it, rather than by providing code. With this technique, the security system will transmit and provide the needed code to other hosts on behalf of the untrusted host. Thus, the untrusted host never explicitly transmits code to other hosts thereby reducing the security threat posed by the untrusted host.

Thus, in accordance with the invention, a jumping application security system is provided wherein the jumping application security system may be a spoke and hub architecture or a peer-to peer network. The jumping application security system comprises a *management and security console* computer that executes instructions for controlling the security of a jumping application and one or more host computers connected to the console computer wherein each host computer executes the jumping application that jumps from host to host during execution. The security console further comprises means for monitoring the security of the jumping application as it jumps between a dispatching host and another host wherein information about the jumping application is communicated to the console computer, means for providing a list of allowable executable programs (or portions of executable programs), means for allowing a host to specify which executable program (or portion of an executable program) to transmit to other hosts, and means for having the security system transmit the specified executable program (or portion of an executable program)to other hosts. A method for jumping application security is also described.

Thus, in accordance with the invention, a computer implemented jumping application security console that maintains the security of a jumping application that is jumping between one or more hosts connected to the security console is provided. The security console comprises a security module that controls the security of a jumping application and a database that contains one or more pieces of code and a description of each piece of code, wherein each piece of code implements a particular behavior. The security module further comprises instructions that replace code from the jumping application that implements a first behavior with a piece of code from the database into the jumping application that implements the first behavior.

In accordance with another aspect of the invention, a computer implemented jumping application security console that maintains the security of a jumping application that is jumping between one or more hosts connected to the security console is provided. The security console comprises means for controlling the security of a jumping application and means for storing one or more pieces of code and a description of each piece of code, wherein each piece of code implements a particular behavior. The security controlling means further comprises means for removing code from the jumping application that implements a first behavior and means for inserting a piece of code into the jumping application that implements the first behavior.

In accordance with yet another aspect of the invention, a computer-implemented method for controlling the security of a jumping application in a jumping application system is provided. In the method, a request is received for a piece of code that implements a particular behavior for a jumping application and the code in the jumping application that implements the particular behavior is replaced with a piece of code that implements the particular behavior into the jumping application so that the jumping application has the particular behavior when it is executed by a host in the jumping application system.

Brief Description of the Drawings

Figure 1 illustrates a typical jumping application;

Figure 2 illustrates an example of the execution of a typical jumping application;

Figure 3 is a diagram illustrating how conventional jumping application systems handle the code of a jumping application;

Figure 4 is a diagram illustrating an example of a jumping application security system in accordance with the invention that improves on current jumping application techniques;

Figure 4A is a diagram illustrating an example of a preferred embodiment of a jumping application security system in accordance with the invention;

Figure 5 is a diagram illustrating the architecture of the preferred embodiment of the jumping application security system; and

Figure 6 is a diagram illustrating the details of the preferred embodiment of the jumping application security system in accordance with the invention.

Detailed Description of a Preferred Embodiment

The invention is particularly applicable to a jumping application system for a client/server type jumping application computer system and it is in this context that the invention will be described. It will be appreciated, however, that the jumping application security system and method in accordance with the invention has greater utility since it may be used for the protection of any computing system and for any jumping application. For example, the inventive system may be used with wireless computing devices (e. g., cell phones, wireless e-mail devices, wireless computer devices and the like), it may also be utilized with peer-to-peer computer systems as well as any other type of computer system that is capable of executing a jumping application. To better understand the invention, a typical jumping application and an example of its execution will be provided.

Figure 1 illustrates a typical jumping application 18 and its operation. In particular, the jumping application may start its execution on a first computer 20. At some point, the jumping application 18 is instructed to move to a second computer 22 and the jumping application jumps (e.g., is communicated to or is sent) to the second computer. Once at the second computer, the jumping application resumes its execution on the second computer. At

some later time, the jumping application is instructed to move to a third computer 24 and the jumping application jumps to the third computer and resumes its execution on the third computer. In this manner, the jumping application can execute on one or more different computers at different times. To understand the concept of a jumping application, an example of a typical jumping application will now be provided.

Figure 2 is a diagram illustrating an example of a typical jumping application 18 and in particular, an intelligent expense report form. In this example, the jumping application facilitates the expense report process by automatically performing some functions. In particular, a salesman at a laptop computer 26 may initially fill out an expense report form and click OK when the expense report is ready. Automatically, the jumping application then sends itself to a manager 's computer 28 for approval by the manager. In this example, the manager finds a problem with the form and returns it to the salesman so that the form automatically sends itself back to the salesman 's computer 26 for an update. Next, the salesman makes the necessary corrections and clicks OK to send it automatically back to the manager 's computer 28. With the further updates, the manager accepts the expense form and clicks "OK'. The jumping expense report form then automatically sends itself to a computer 30 in the administration department. The jumping expense form then executes on the administration computer and updates a database 32 with the new information in the expense form. Next, the jumping expense report automatically sends itself to a computer 34 of the accountant. The jumping expense report then automatically starts to execute on the accountant 's computer and notifies the accountant that a check is needed so that the accountant can cut the check for the salesman. Thus, the jumping application has automated much of the expense report submission process so that the people involved in the process do not have to worry about ensuring that the expense report is approved. Now, the jumping application security system in accordance with the invention will be described.

Figure 3 illustrates earlier techniques for managing the code of a jumping application. With earlier techniques, a host computer 102, Host 1, would instantiate a jumping application 112, and then later dispatch it to the second host 104, Host 2. In this example, Host 1 is untrusted, so the jumping application system simply strips all of the code from the jumping

application. The jumping application system would then forward the jumping application to Host 2 without any code since the code has been stripped from the jumping application. Later, the jumping application would be dispatched to Host 3. In this example, Host 2 is trusted, so any code included with the jumping application would be transmitted to Host 3, unmodified. This would be repeated on each jump, and the system would determine if the code should be transmitted or not depending on whether or not the sending host is trusted. Thus, with previous jumping application techniques, a jumping application might arrive at a host without any code. In this case, the jumping application would need to retrieve any needed code from some trusted source, which may or may not be available. If the trusted source is not available, an error condition will result and the jumping application will stop its proper operation. Now, a jumping application security system in accordance with the invention that overcomes the limitations of the typical jumping application security system will be described.

Figure 4 illustrates an example of a jumping application security system 128 in accordance with the invention. In this security system in accordance with the invention, the jumping application system can provide code to a jumping application based on a description of the desired behavior provided by the jumping application. Thus, the security of the jumping application is maintained since all code is being provided by the security system (which is trusted) yet the code required by the jumping application is being provided to the jumping application so that the jumping application may continue its operation.

In this example, a jumping application 124 is instantiated on the first host 126, Host 1. In accordance with the invention, the jumping application (through Host 1 communicating with the security system 128) describes to the security system 128 what behavior it wants. In other words, the jumping application requires some code in order to execute/implement a particular function and describes that function to the security system. In this example, the jumping application wants the behavior of an expense report. Later, the jumping application is dispatched to a second host 130, Host 2. Prior to sending the jumping application to Host 2, in accordance with the invention, the security system 128 will remove any code provided by Host 1, such as any expense report code, and replace it with its own code that provides the

"expense report" behavior/functionality. Then the security system 128 forwards the jumping application to the second host 130, Host 2 with the security system's "expense report" code so that the jumping application with the "expense report" functionality (in this example) continues its execution on Host 2. More generally, the security system inserts its code for the desired functionality into the jumping application so that the jumping application with the desired functionality continues its execution on the next host. The above process is repeated for each jump made by the jumping application. Thus, with each jump, the jumping application security system replaces the transmitted code with its own code for the desired behavior/functionality so that the jumping application has the code necessary to implement the desired functionality while ensuring the security of the jumping application since no host must be trusted in accordance with the invention. Now, an example of a jumping application security system in accordance with the invention will be described.

Figure 4A is a diagram illustrating a client/server jumping application security system 128 in accordance with the invention. As described above, this architecture of the jumping application security system is an example of an implementation of the jumping application security system as the jumping application security system may be implemented using various different architectures. In this example, the system 128 may include a server computer 52 and one or more host computers 54, such as Host 1, Host 2 and Host N, that may be connected to the server computer by a computer network 56, such as a wide area network, the Internet, a local area network, the World Wide Web, a telephone line and a modem or the like. The computer network permits the server and hosts to communicate data between each other using well known protocols and data formats. Each host may be a typical computer system that includes the well known computing resources, such as at least a CPU and a memory, for executing a software application such as a jumping application. Each host may be a personal computer, a laptop, a server computer, a work station, a personal digital assistant, a Pocket PC computer, a cellular phone, etc.. with sufficient computing power to execute a jumping application.

The server 52 may include one or more CPUs 58 and a memory 60 along with a persistent storage device (not shown) for permanently storing one or more software

applications or modules that may be executed by the CPU by loading the software applications or modules into the memory. The server may also include well known input/output devices that are not shown as well as a device for connecting to the computer network 56, such as a modem. DSL modem, etc.. The server may also include a database 62 that stores one or more jumping applications along with information about the jumping applications as described below. The database 62 may further include one or more different pieces of code that implement one or more different functions/behaviors wherein the pieces of code may be inserted into a jumping application as described above. When the server computer is being utilized as the jumping application security system 50, the memory of the server has a jumping application controller module 140 (also known as a management and security console) stored in it that, when executed by the CPU, control the security of the one or more jumping application(s) in the jumping application system and the hosts as described below. In a preferred embodiment, the jumping application controller 64 may be one or more software application or modules, but the controller may also be implemented using hardware.

In a preferred embodiment, the jumping application controller 64 may include a security software module 66 and a communications software module 68. The security module may control the operation of the jumping application security system and maintain the security of the system, such as by inserting the code with the appropriate behavior into a jumping application upon request from the jumping application. The communications module may control the communications with the hosts associated with/connected to the jumping application security system, such as by receiving the request for code with a particular behavior from a host and by sending the code with the particular behavior to the jumping application. Thus, the combination of the security system software may solve the problems with typical jumping application systems so that: 1) the security of the jumping application system is maintained and a host cannot introduce code into the jumping application; and 2) each jumping application receives the code necessary to implement a particular behavior.

Figure 5 illustrates the architecture of a preferred embodiment of a jumping application system 100 in accordance with the invention. In this architecture, there is the Management and Security Console 140 (MaSC) which forms the hub of a spoke-and-hub

arrangement. In this arrangement, the hosts 142, 144, 146, 148, and 150 never communicate directly with each other. Instead, the hosts communicate only with the MaSC 140. This in turn implies that on each jump, each jumping application must pass through the MaSC 140 so that the MaSC controls the jumping application and its security. In the preferred embodiment of the invention, a trusted party or system administrator has access to the MaSC 140, where this trusted party can provide code (for each particular behavior/functionality) which is known to be safe. This arrangement of the preferred embodiment allows the MaSC 140 to substitute known safe code on each jump. This code can come from the known safe code provided by the trusted party at the MaSC.

Figure 6 illustrates the details of the preferred embodiment of the jumping application system 100, based on the architecture of Figure 5, that includes the management and security console 140. In this example, the jumping application 124 is instantiated on Host 1 126, and later dispatched to Host 2 130. The process by which the jumping application 124 jumps between the hosts and receives safe code from the MaSC 140 in accordance with the invention will now be described. Each of the steps of the preferred embodiment described below may be implemented, in the preferred embodiment, as one or more instructions (computer code executing on the management and security console and/or a host computer) that implement the operations described below. In accordance with the invention, these instructions may be written in various well known programming languages or other programming languages as the invention is not limited to instructions written in any particular programming language.

In accordance with the invention, the MaSC 140 contains (in the database 62 shown in Figure 4A) a list of previously developed programs (safe programs) and the software code for each program. Each of these programs is suitable as a jumping application or portion of a jumping application. These programs (pieces of software code) are supplied to the system administrator (and the MaSC 140) from trusted parties so that each piece of code is known to be safe to use in a jumping application. Each of these programs has a description as well as the associated code as shown. In step 2, Host 1 126 queries the MaSC 140 for a list of descriptions of the available programs on the MaSC. Each item in the list is a description (including the particular behavior/functionality) of one of the programs on the MaSC. In step

3, Host 1 selects the desired program(s) from the list of descriptions downloaded in Step 2, which are required to implement a particular behavior(s) within the jumping application. Host 1 then sends a message to the MaSC requesting the executable piece(s) of code. The executable code is specified using a description from the list of Step 2. In step 4, the MaSC 140 provides the specified code to Host 1 and records what code was provided for the jumping application 124 at Host 1 so that the MaSC 140 maintains a list of the code being provided to each jumping application.

In step 5, Host 1 uses the code received in Step 3 by inserting that code into the jumping application and instantiating the jumping application 124 with the inserted code. In step 6, the jumping application is dispatched to Host 2 130 in accordance with the code contained in the jumping application. In step 7, in accordance with the preferred embodiment, the jumping application is sent to the MaSC, prior to being sent to Host 2 130, where the MaSC then removes any code from the jumping application. In step 8, using the information recorded in Step 4, the MaSC determines which program(s)/code to supply with the jumping application, and adds those piece(s) of code to the jumping application. In step 9, in accordance with the preferred embodiment, the jumping application is then forwarded to Host 2 130 from the MaSC 140 and the jumping application arrives at Host 2. In step 10, the jumping application resumes execution on Host 2, using the code supplied by the MaSC in Step 8 so that the jumping application has the required behavior (using the code supplied by the MaSC 140), but unsafe code for the behavior is not inserted into the jumping application. Thus, this arrangement of the preferred embodiment allows the MaSC to substitute known safe code on each jump.

While the foregoing has been with reference to a particular embodiment of the invention, it will be appreciated by those skilled in the art that changes in this embodiment may be made without departing from the principles and spirit of the invention, the scope of which is defined by the attached claims.